README file to run Whole Genome Samples in COMPUTE1 – by Matt Johnson

V1- date: December 2, 2022

This WXS-Pipelines repository is designed for processing whole genome and exome sequencing data on WUSTL's compute1 HPC cluster. It can be found at https://github.com/NeuroGenomicsAndInformatics/WXS-Pipelines.

**Inputs data:**

1. A text file to be used as an argument for the pipeline script with all the samples that need to be processed in FULLSMID format. The FULLSMID format refers to SAMPLE\_ID^BARCODE^SEQPROJECT. Samples should be separated by whitespace. Spaces, tabs, or newlines are acceptable.
   1. ex. MAP\_10040.GRAYS^2102557103^20202110\_MGI\_WGS\_GRAYS
2. Once data is downloaded from GLOBUS or BOX, it is staged in storage 1 Active with the following structure:
   * /storage1/fs1/cruchagac/Active/$USER/c1in/${FULLSMID}/${FLOWCELL}.${LANE}\_${READNUM}.fastq.gz
     1. The pipeline has a limited ability to pull flowcell and lane data from the fastqs, but the filename must end in \_${READNUM}.fastq.gz. Providing the full name with flowcell and lane is still recommended.
   * ex. /storage1/fs1/cruchagac/Active/matthewj/c1in/MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS/HLFK3DSXY.1\_1.fastq.gz
   * /storage1/fs1/cruchagac/Active/matthewj/c1in/MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS/HLFK3DSXY.1\_2.fastq.gz
3. Fastqs need specific filenames with the flowcell and lane for RGfile generation, but bams and crams can be named however as long as they're in a FULLSMID-named directory.

**Outputs files:**

Each set of output files can be found in/storage1/fs1/cruchagac/Active/$USER/c1out/${FULLSMID}/

1. Aligned, sorted, duplicate-marked crams for each sample
   * ex. MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram
   * MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram.crai
2. A gVCF for the sample
   * ex. MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.snp.indel.g.vcf.gz
   * MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.snp.indel.g.vcf.gz.tbi
3. Coverage reports
   * ex. MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram.rawwgsmetrics.txt
   * MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram.wgsmetrics.txt
   * MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram.wgsmetrics\_paddedexome.txt
4. VerifyBamID report
   * Ex. MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram.vbid2.Ancestry
   * MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.aln.srt.mrk.cram.vbid2.selfSM
5. Variant Calling Metrics
   * ex. MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.snp.indel.g.vcf.gz.vcfmetrics.variant\_calling\_detail\_metrics
   * MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.snp.indel.g.vcf.gz.vcfmetrics.variant\_calling\_summary\_metrics
6. Various outputs, logs, and reports
   * ex. MAP\_10040^2102557103^20202110\_MGI\_WGS\_GRAYS.env - The environment variables used to process the sample

**STEPS TO RUN PIPELINE**

1. Stage data to storage1
   * Use the structure defined above in "Inputs"
   * There is also a script in the GitHub repository called prepareDataForStaging.bash that demonstrates how one might stage with a script and can be modified.
2. Log into compute1 (if you aren't already)
   * ssh [USER@compute1-client-1.ris.wustl.edu](mailto:USER@compute1-client-1.ris.wustl.edu)
   * NOTE: The above uses client 1, but there are 4 client nodes numbered 1-4
3. Create a workfile
   * Any text file should work. It should contain a list of FULLSMIDs that correspond to the folder names of the staged data.
   * Spaces, tabs, or new lines separating each sample should work and have all been tested.
   * Running a list command to get the folders is a quick way of doing this.
   * **ls /storage1/fs1/cruchagac/matthewj/c1in/\*${SEQUENCING\_PROJECT} > workfile.txt**
4. Get the pipeline code
   * The pipelines are publicly available via github. You can get them by navigating to where you want the folder and entering:
   * git clone <https://github.com/NeuroGenomicsAndInformatics/WXS-Pipelines.git>
5. Run the pipeline

* Navigate to the folder that was just added.
  + cd </path/to/>WXS-Pipelines
* Call the pipeline script. Note that the names refer to the input filetypes.
* ex. bash wgsfqspipeline.bash </path/to/>workfile.txt

1. Collect outputs when done
   * The output files will show up in /storage1/fs1/cruchagac/$USER/c1out/$FULLSMID
   * The pipeline repository also includes a transferOutputFiles.bash script to copy files back to storage1. The input could also be the same workfile.

The steps in the processing pipeline are below:

**1. Prepare fastq files (if necessary)**

* Crams are first reverted with RevertSam to remove all prior tags to ensure clean processing from scratch. The reverted data is piped to SamToFastq.
* SamToFastq separates the data into paired-end fastq files by read group. This step ensures that all samples are processed identically by creating a common state for the raw data. This common state is read group-separated fastq files.

**2. Align and sort fastq reads**

* Reference files for the alignments, and the entire pipeline, were taken from GATK’s resource bundle and were downloaded from the Google bucket on August 12 of 2019 and are GRCh38.
* The reads were aligned with either bwa-mem2 on CPUs or NVIDIA’s Clara Parabricks version 3.6.1 on GPUs. Both of these perform the same function and produce the same outputs as bwa-mem version 0.7.15 and 0.7.17. The alignments were done in alternate-aware mode with alternate contigs in the GATK resource bundle. Bwa-mem2 also requires index files that were locally generated from the resource bundle reference fasta.
* The reads are coordinate-sorted with GATK’s SortSam tool. This tool is internally a Picard tool.

**3. Mark duplicates**

* The marking of duplicate reads was done with either GATK 4.2.6.1 or Parabricks 3.6.1. These tools perform the same functions and lead to the same outcomes, but the Parabricks version is able to take advantage of GPUs to massively parallelize the job. In the case that a GPU wasn’t available, GATK performed the work on CPUs.
* We are able to utilize the read data in cram format beyond this point. After marking duplicates, the read data is stored in cram format and those are used to process the data further.

**4. Recalibrate bases**

* GATK BaseRecalibratorSpark was utilized to perform base recalibration in parallel. While providing identical results, this tool was able to recalibrate base and mapping qualities an order of magnitude faster than its serial counterpart.
* Mills-Gold
* dbSNP 138
* 1K genome Phase 1

**5. Call variants with HaplotypeCaller**

* Parabricks 3.6.1 was utilized for variant calling on the genomes. Parabricks 3.6.1 performs the HaplotypeCaller algorithm from GATK 4.1.0 with near identical results. It also saves the step of applying the BQSR values to a bam. Because of the scale of data being processed, Parabricks’ implementation was the preferred option because others were prohibitively slow.

**QC components:**

1. Mean coverage

* GATK 4.2.6.1’s CollectRawWgsMetrics and CollectWgsMetrics provided coverage data for the samples. Summary data was generated for coverage at base and mapping quality of 3 and 0, respectively, by CollectRawWgsMetrics. CollectWgsMetrics represents a higher quality threshold for the data with base and mapping quality minimums of 20, corresponding to 99% certainty.
* Coverage data was also collected with CollectWgsMetrics over only the padded exome.

2. Freemix

* Freemix was calculated with Griffan Lab’s VerifyBamID, an updated version of the original. These are used to measure the probability of contamination.
* The data used for this analysis come from the 1000 Genomes – Phase 3 data release. The files used can be found in the repository for the tool.

3. TiTv Ratio

* GATK 4.2.6.1’s CollectVariantCallingMetrics was used to capture a variety of data from the gVCFs. Of note are the Percent SNPS in dbSNP, dbSNP TiTv Ratio, and Novel TiTv Ratio that we use to measure the total TiTv ratio for the data.

4. Annotations of key genes

* SnpEff/SnpSift 5.1 annotate APP, PSEN1, PSEN2, GRN, MAPT, TREM2

5. Stats File

* A csv with a header line and a data line.
* Key QC stats from above
* Counts of annotations on MANE transcripts from the key genes above
* Counts of HIGH and MODERATE effect annotations from the key genes above
* The HIGH and MODERATE effect variants from the key genes above

**MAKING CHANGES TO THE SCRIPTS**

While the repository has several versions of scripts that should work for most in Cruchaga Lab with compute1 access, there may be a need to modify these scripts. Dockers are more involved to change, so they’ve been written without any hard-coded dependencies. Most scripts offer a means to change which user accounts are used by the job submissions with a set of variables at the beginning of the script. These include the pipeline scripts and most of the wrapper scripts.

1. The variables, generally near the top of scripts, are ${STORAGE\_USER}, ${SCRATCH\_USER}, and ${COMPUTE\_USER}. These variables relate to which user’s storage allocation you’ll be using, which user’s scratch space you’ll be using, and which compute user’s cpus you’ll be using. Most of the time these will all be the same user, but it is more flexible to separate them.
2. To change the references or reference locations, there is a references.env file that is used in the makeSampleEnv.bash scripts that are added to the environment for the pipeline.